!!! Do *not* put the assignment on a public github, bitbucket, etc. repository.

!!! If you make the assignment public, your job application will be declined immediately

Background

Imagine you are part of a team working on an Optical Character Recognition (OCR) system, that is a system that can read text from images. A well known open source system for OCR is tesseract, which implements traditional and Deep Learning (DL)-based algorithms.

Here, we are interested in building a web-service that accepts images, uses tesseract to do OCR in the background, and returns the text from the image.

Assignment

Please implement a web-service that implements the following http-based API:

REST API

POST /image-sync

Accept the image as a b64-encoded payload data:

```
curl -XPOST "http://localhost:5000/image-sync" \
    -d '{"image_data": "<b64 encoded image>"}'
```

It should return a response, content-type `application/json`:

```
{
    "text": "<recognized text>"
}
```

POST /image and GET /image

The `POST /image-sync` API is actually not so good, because OCR may take some time, especially for large images. It is generally bad practices to have http services which takes multiple seconds or worse to respond. We thus ask to add another API that works as follows. The input is exactly the same as for `/image-sync`:

```
curl -XPOST "http://localhost:5000/image" \
    -d '{"image_data": "<b64 encoded image>"}'
```

But instead, it should return a response, content-type `application/json`:

```
{
    "task_id": "<task id>"
}
```

The task id can then be used to retrieve the OCR text with the `GET /image`:

```
curl -XGET "http://localhost:5000/image" \
    -d '{"task_id": "<task id as received from POST /image>"}'
```

and return a response, content-type `application/json`:

```
{
    "task_id": "<recognized text>"
}
```

if the task is done, or:

```
{
    "task_id": null,
}
```

In which case it is assumed the task is not finished yet.

Requirements

More concretely, we would like you to implement:

1. A web service that implements the above API.
2. For the "async" API (`POST /image` and `GET /image`), it is expected that the actual OCR is run through some background worker(s), that do not block the web server implementing the web API.
3. For the actual OCR, we recommend you to use tesseract.

One way to use tesseract is to use a Ubuntu docker container defined as follows:

```
# Dockerfile
FROM ubuntu:18.04

RUN apt-get update && apt-get install -y tesseract-ocr-eng
```

Then you can get OCR as follows, from inside the docker container:

```
# <image filename> must be a supported format, such as tiff or png. If
# successful, you should have a file output.txt containing the recognized
# text. Be careful that the command line `tesseract` expects the .txt
# extention *not* to be included.
$ tesseract --psm 1 --oem 1 <image filename> output
```

As an image file example, you can use this .tif file.

Expectations

The code must be written in one or more of the following languages: Python, Golang, NodeJS. If you use another language, it may take longer for us to review it, as we may not have many reviewers familiar with it. If in doubt, just ask us.

The code should be written and packaged in a state you consider ready for use. The assignment will be judged on correctness, but also usability, design.

It is particularly important that the assignment is sent such as we can actually run it easily, and try some requests. We will not spend one hour to try to run your assignment.

Non requirements

We do not expect the following to be implemented:

1. High Availability. We do not expect the assignment to be robust against machine failure, etc.
2. Scalability. We do not expect the assignment to be scalable to large throughput.
3. Writing your own OCR engine. We expect candidates to use tesseract.

Especially for points 1 and 2, more experienced engineers are expected to be able to answer questions related to those points during the technical interview, but we do not expect those points to be implemented.

Going beyond

If the above task sounds too simple, please consider how to expand the exercise, keeping in mind the context of the exercise. For example, how would you change the architecture to scale ? How would you improve the API if you wanted to support sending batches of images and being notified of task readiness instead of the poll API ?

If you have an machine learning/deep learning background, you may also think about how to actually implement the OCR part yourself instead of using tesseract.

How to submit the assignment

If you use source control (such as git, hg, etc.), simply archive the repository into a tarfile or similar format. Please make sure not to publish your repository anywhere public, as this would immediately disqualify you.

Otherwise, put all the code, data, instructions, etc. together in a tarfile or similar format. Send the tarball as an attachment to the following address: mercari-jp-ml-assignments@mercari.com.

For the tech interview following the assignment, please make sure to do one of the following:

1. If the interview is onsite, please come with your assignment ready on your machine. If you do not have a machine ready for the assignment, please let us know in advance and we will prepare one.
2. If the interview is remote, make sure to have your assignment ready so that the code can be shared through screen sharing (hangout or similar video-conference software).